Goddard Procedural Requirements (GPR)

DIRECTIVE NO.	GPR 7150.4	**APPROVED BY Signature:**	***Original Signed By***
EFFECTIVE DATE:	September 24, 2012	**NAME:**	Judith N. Bruner
EXPIRATION DATE:	September 24, 2017	**TITLE:**	Director, Safety and Mission Assurance

COMPLIANCE IS MANDATORY

Responsible Office: Code 300/Safety and Mission Assurance (SMA)

Title: Software Safety and Software Reliability Process

TABLE OF CONTENTS

PREFACE

P.1 PURPOSE

This directive presents Goddard Space Flight Center's (GSFC) implementation of the National Aeronautics and Space Administration (NASA) Software Safety Standard, NASA-STD-8719.13 for GSFC-managed projects. To the extent possible, this GPR avoids duplication with other NASA and center directives and requirements. Some requirements have been directly incorporated from the NASA Software Safety Standard, NASA-STD-8719.13 for clarity.

P.2 APPLICABILITY

a. This software safety and software reliability directive shall be applied to all safety-critical software and mission-critical software (see definitions in Appendix A) acquired or developed by the NASA GSFC (Greenbelt and Wallops Flight Facility (WFF)) in support of developmental and operational projects.

b. For existing projects, this GPR applies to the project phases yet to be completed as of the effective date of this GPR to the extent determined on a case-by-case basis (see section 1.3 for details).

P.3 AUTHORITY

a. NPR 7150.2, NASA Software Engineering Requirements

b. NASA-STD-8719.13, Software Safety Standard

P.4 APPLICABLE DOCUMENTS

a. NPR 1441.1, NASA Record Retention Schedules

b. NASA-STD-8739.8, NASA Software Assurance Standard

c. NASA-GB-8719.13, NASA Software Safety Guidebook

d. GPR 1400.1, Waiver Processing

P.5 CANCELLATION

None

P.6 SAFETY

None

P.7 TRAINING

a. Software Safety Engineers (SSEs) and Software Reliability Engineers (SWREs) shall have expertise in software and systems engineering.

b. All software safety and software reliability practitioners shall have a working knowledge of system hazards and failures, software causes or contributors to hazards and mission failures, and the role of safety-critical and mission-critical software in GSFC systems.

c. All software safety and software reliability practitioners shall also be required to take the System for Administration, Training, and Educational Resources at NASA (SATERN) Software Safety for Practitioners course and the SATERN Introduction to Software Reliability course, respectively.

d. Any additional training requirements (e.g., select discipline courses via the Safety and Mission Assurance Technical Excellence Program (STEP)) for software safety, reliability and assurance practitioners shall be specified by the project and/or the respective assurance organization.

P.8 RECORDS

a. Software Assurance personnel shall maintain results from Software Safety and Software Reliability process and product assessments conducted for each project under development.

Record Title	Record Custodian	Retention
Software Safety and Software Reliability Process and Product Assessment Results	Code 320, Software Assurance Tracker (SwAT) System	*NRRS 8/36.5/C/1 - Handle as permanent.

**NRRS – NASA Records Retention Schedules (NPR 1441.1)*

b. Results of safety, reliability and hazard analyses shall be maintained as part of the Project/Program/Mission Configuration Management System, as per NPR 7123.1, NASA Systems Engineering Processes and Requirements.

Record Title	Record Custodian	Retention
Instrument Preliminary Hazard Analyses (PHAs) and Safety Assessment Reports, Project PHAs and Safety Data Packages (SDPs), Project and Instrument Reliability Assessment Results (i.e., Failure Modes and Effects Analyses (FMEAs), Probabilistic Risk Assessments, and Fault Tree Analyses (FTAs))	Project Office	*NRRS 8/103 - Temporary. Destroy/delete between 5 and 30 years after program/project termination.

**NRRS – NASA Records Retention Schedules (NPR 1441.1)*

P.9 MEASUREMENT/VERIFICATION

a. Software Safety and Software Reliability Process and Product Assessment(s) shall be performed for each project by Software Assurance Engineers within Code 320 or Code 803 to measure progress towards fulfilling the requirements of this GPR, as stated in sections 7.1, 7.2, 7.3 and 7.4.

b. If the above-mentioned Software Safety and Software Reliability Process and Product Assessment(s) indicates that one or more of the requirements contained within this GPR cannot be met, then a waiver/deviation package shall be prepared and submitted in accordance with the requirements of GPR 1400.1, Waiver Processing (see section 7.5 for details).

PROCEDURES

In this document, a requirement is identified by "shall," a good practice by "should," permission by "may" or "can," expectation by "will," and descriptive material by "is."

1. Project Manager

The Project Manager (PM) is responsible for ensuring sufficient resources and personnel have been allotted to fulfill the software safety and software reliability requirements and activities. The PM is also responsible for assuring that software safety and software reliability are included as part of the entire system lifecycle.

1.1. For WFF Projects, the PM shall work directly with Code 803 to determine who shall fulfill the roles and responsibilities of the Chief Safety and Mission Assurance Officer (CSO), Project Safety Manager (PSM), Software Assurance Engineer (SAE) and Project Reliability Engineer (RE), as outlined in sections 2, 3, 7 and 8 of this document, respectively.

1.2. The PM shall work with the CSO to assure the provision of adequate resources, including trained personnel, schedule time, tools, and budget to facilitate the following tasks:

a. Implementation of a process or mechanism to document, trace, communicate, and close software safety and reliability concerns, including those that result from design reviews, safety analyses or reliability analyses, with concurrence of the safety or reliability personnel, respectively.

b. Implementation of project, ground, and facility risk management processes that capture, address and manage risks associated with or affecting software safety or reliability.

1.3. The PM shall work with the CSO to determine the applicability of this GPR to the remaining phases of a project that exist as of the effective date of this GPR.

2. Chief Safety & Mission Assurance Officer

The CSO is responsible for assuring overall execution of the software safety and software reliability program on behalf of Code 300 throughout the entire software life cycle and negotiating resources and activities with the Project Manager and respective Branch/Discipline Leads. The CSO assures coordination of software safety and software reliability activities with all other Safety & Mission Assurance (SMA) disciplines.

2.1. The CSO shall include software safety requirements and software reliability requirements in the Mission Assurance Requirements (MAR). This will be done by working with the SSE, SWRE and SAE.

2.2. The CSO shall assure planning, integration, execution and reporting of software safety and software reliability activities, including synchronization with related program/project milestones.

2.3. The CSO shall assure that software safety and software reliability activities are performed during the software development life cycle.

2.4. The CSO shall coordinate software safety and software reliability tasks and associated funding levels with the PSM and Project RE, respectively, as well as the SAE.

2.5. The CSO shall assure that risks which could manifest themselves as a system hazard or impose a system failure are captured in the hazard analyses or the reliability analyses, respectively.

2.6. The CSO shall communicate software safety and software reliability risks directly to SMA management and Project Management for resolution within the project.

2.7. The CSO shall work with Code 320 or Code 803 to designate personnel that are responsible for the software safety and software reliability disciplines of the project, program, or facility.

2.8. The CSO shall ensure that all personnel performing software safety and software reliability roles on the project are trained to perform their respective duties as documented in this GPR.

2.9. The CSO shall document all variances to the requirements contained in this GPR.

2.10. The CSO shall provide a means to resolve conflicts related to software safety and software reliability requirements or processes.

2.11. The CSO shall assure the proper implementation of the requirements listed in sections 1.2 and 1.3 of this GPR.

2.12. If an SSE does not exist for a project, the CSO shall work with Code 320 or Code 803 and the PM to determine who will fulfill the responsibilities of the SSE.

2.13. If an SWRE does not exist for a project, the CSO shall work with Code 320 or Code 803 and the PM to determine who will fulfill the responsibilities of the SWRE.

3. Project Safety Manager

The PSM is responsible for addressing software safety planning as part of a System Safety Program Plan. The PSM is responsible for conducting the safety analyses (e.g., PHA) and documenting software

causes, controls, and verifications in the associated hazard reports which are incorporated into the applicable SDPs. The PSM works with systems engineering, software engineering, software assurance, reliability engineering and project management to identify and characterize the safety risks associated with software.

3.1. The PSM shall classify software as safety-critical if the software can cause or contribute to a hazard or provides a control(s) to a hazard.

3.2. The PSM shall assure:

a. All software-related hazard causes have been identified.

b. All software-related hazard controls have been identified.

c. All software safety-critical requirements and associated software design elements have been identified and tracked.

d. All software safety-critical requirements and associated software design elements have been successfully validated.

e. All software safety-critical requirements and associated software design elements have been properly verified.

f. All discrepancies in safety-critical software have been dispositioned.

3.3. The PSM shall assure that all off-the-shelf and reused software, including software present in Ground Support Equipment are evaluated as part of the safety analyses.

3.4. The PSM shall ensure that the acquired or developed system is evaluated for impacts to software-related hazard causes and software-related hazard controls resulting from changes to the associated hardware or software (e.g., change requests, discrepancy reports, waivers, problem reports, etc.).

3.5. The PSM shall ensure that the above-mentioned evaluations are performed (at a minimum) at major lifecycle reviews and when significant changes are made to the system.

3.6. The PSM shall communicate software safety risks directly to the Software PDL and elevate risks to the Systems Engineer and/or Project Management, if necessary, for resolution within the project.

3.7. The PSM shall, in parallel to the above-mentioned task (section 3.6), communicate software safety risks directly to the CSO and SMA management.

3.8. Utilizing the safety analyses, such as the PHA and subsystem hazard analyses, the PSM shall provide input to the Software PDL regarding which failures to test for and the level of failure combinations to include.

3.9. The PSM may tailor this GPR for software that either can cause a critical or catastrophic hazard, but is controlled via the use of 2 or 3 independent, non-software-related inhibits, respectively, or aids in controlling a hazard, but only plays a supplementary role to the required inhibit scheme.

3.10. Note that the software mentioned in the above paragraph (section 3.9) is likely to be low risk. As a result, even though such software is still considered to be safety-critical and the associated requirements are tagged as such, this GPR may be tailored such that the level of effort for this software throughout the remainder of the project lifecycle is focused upon ensuring that any changes made to the software or hardware neither allow the software to bypass nor affect the capability of the non-software-related inhibit scheme.

3.11. The PSM shall coordinate software safety tasks with the CSO.

3.12. The PSM shall provide input to the CSO regarding the funding levels associated with the software safety tasks.

4. Systems Engineer

The Systems Engineer (SE) is responsible for assuring system level verifications of safety-critical software requirements and mission-critical software requirements not handled at the product level.

4.1. Safety and reliability engineers are reliant upon the SE, along with the requirements management process for tasks such as the following:

a. Translating Mission Objectives and Success Criteria into functional and performance requirements that can be used to design and validate each of the associated subsystems.

b. Flow of safety and reliability analysis into the requirements management process, in order to improve the design of the associated subsystems.

c. Synchronization of designs, with the Mission/Operations Concept, which is used to extract information such as Mission Phases, Modes, and the sequence of operations associated with Critical Events, all of which are key inputs to “Test as you fly” approach and the corresponding safety and reliability analyses.

4.2. Using safety and reliability analyses, such as the PHAs, subsystem hazard analyses, FMEAs, FTAs and the associated risk assessments (including likelihood and consequence assessments), the SE shall determine which failures to test for and the level of failure combinations to include. This will be accomplished by working with the PSM and RE.

4.3. The SE shall ensure that system testing verifies correct and safe operations in conjunction with system hardware and operator inputs in all anticipated operational and off-nominal configurations. This includes testing in the presence of failures and faults including software, hardware, input, timing, memory corruption, communication, and other failures.

5. Product Development Lead/Software Engineer

The Software Product Development Lead (PDL)/Software Engineer, hereafter referred to as the Software PDL, is responsible for implementing and verifying the safety-critical software requirements and mission-critical software requirements. The Software PDL is responsible for informing the SSE and SAE of any changes to safety-critical software requirements that could potentially affect the safety analyses prior to approval of the associated change. The Software PDL is responsible for informing the SWRE and SAE of any changes to mission-critical software requirements that could potentially affect the reliability analyses prior to approval of the associated change.

5.1. The Software PDL shall use the system hazard analyses and software safety analyses to create new, or identify existing software requirements necessary to control any hazards where software is a potential cause or contributor, or enable software to be used as a hazard control. Such requirements are safety-critical software requirements. This will be achieved by working with the PSM.

5.2. The Software PDL shall use the reliability analyses to create new, or identify existing software requirements necessary to mitigate or resolve any failures where software is a potential cause or contributor, or enable software to be used as a failure mitigation. Such requirements are mission-critical software requirements. This task will be completed by working with the RE.

5.3. The Software PDL shall determine the modes or states of operation under which the safety-critical software requirements and mission-critical software requirements are valid, and any modes or states in which they are not applicable. This will be done by working with the PSM and RE.

5.4. The Software PDL shall ensure that any safety or reliability related constraints between the hardware and software, as documented by the Hardware PDL (ref. section 6.1), are consistent

with the appropriate software requirements specification(s). This will be done by working with the Hardware PDL, SSE and SWRE.

5.5. The Software PDL shall distinctly identify all generic and specific requirements that are either safety-critical or mission-critical software requirements as such in the software requirements specification. This will be accomplished by working with the SSE and SWRE.

5.6. Using the safety and reliability analyses, such as PHAs, subsystem hazard analyses, FMEAs, and FTAs, the Software PDL shall determine design features to prevent, mitigate, or control software-related failures and faults. This will be achieved by working with the PSM and RE.

5.7. The Software PDL shall clearly identify all safety-critical and mission-critical software design elements in the design documentation. This task will be implemented by working with the SSE and SWRE.

5.8. The Software PDL shall distinctly identify all commands, data, telemetry, input sequences, options, error message descriptions, corrective actions and other items that are either safety-critical or mission-critical in operational documentation, including user manuals and procedures. This will be done by working with the SSE and SWRE.

5.9. The Software PDL will ensure that all functional safety-critical and mission-critical software requirements and safety-critical and mission-critical software elements are verified by testing.

5.10. Based upon inputs received from the PSM and RE and the results of the safety and reliability analyses (e.g., PHAs, subsystem hazard analyses, FMEAs, FTAs), the Software PDL shall determine which failures to test for and the level of failure combinations to include.

5.11. The Software PDL shall assure that system testing verifies the correct and safe operation of the software in conjunction with system hardware and operator inputs in all anticipated operational and off-nominal configurations. This includes testing in the presence of failures and faults including software, hardware, input, timing, memory corruption, communication, and other failures.

5.12. The Software PDL shall identify all project tools (e.g., compilers, simulators and automatic code generators) that could potentially impact safety-critical and mission-critical software, the degree of impact, and mitigation strategies in the appropriate project plan.

5.13. The Software PDL shall assess any changes to project tools (e.g., compilers, simulators) that could influence known or introduce new hazards or mission failures. This will be accomplished by working with the SSE, SWRE, SAE, PSM and RE.

5.14. The Software PDL shall identify and assess software changes and discrepancy reports related to safety-critical software. This will be achieved by working with the SSE, SAE and PSM.

5.15. The Software PDL shall identify and assess software changes and discrepancy reports related to mission-critical software. This task will be completed by working with the SWRE, RE and SAE.

5.16. The Software PDL shall make design documents/artifacts, requirements, operational documentation, databases, tests, source code, test plans, and test results available to the SSE and the SWRE.

6. Product Development Lead/Hardware Engineer

The Product Development Lead/Hardware Engineer, hereafter referred to as the Hardware PDL, is responsible for implementing and verifying any hardware constraints that could affect the ability to successfully implement any safety-critical or mission-critical software requirements. The Hardware PDL is responsible for ensuring that the above-mentioned constraints and any associated changes are documented and communicated to the Software PDL and either the SSE and PSM or SWRE and RE.

6.1. The Hardware PDL shall include any safety or reliability related constraints between the hardware and software in the appropriate specification(s) and/or requirements document(s). That is, when the software and hardware work together to perform a safety-critical or mission-critical function, their roles, precedence, and failure modes, are documented and understood. This will be done by working with the Software PDL, along with the SSE and PSM or the SWRE and RE.

7. Software Assurance Engineer

The SAE performs software safety and software reliability product and process assessments to verify compliance to the software safety and software reliability processes and reporting any associated non-conformances or risks to all relevant stakeholders. The SAE is responsible for analyzing the software artifacts to ensure that all software safety elements are traceable back to system hazards and that all mission-critical software elements are traceable back to the reliability analysis. The SAE will assist in analyzing software changes and software anomalies for safety-related and reliability-related impacts.

7.1. The SAE shall perform software safety and software reliability process and product assessments to ensure compliance with this GPR.

7.2. The SAE shall specify the type, number and relative schedule of software safety and software reliability process and product assessments in the Software Assurance Plan.

7.3. Based on the results of the software safety and software reliability process and product assessments, the SAE shall create an assessment checklist indicating compliance or non-compliance with each of the requirements in this GPR.

7.4. The SAE shall maintain the above-mentioned checklist throughout the development lifecycle.

7.5. If the aforementioned checklist indicates that one or more of the requirements in this GPR cannot be met, the SAE shall prepare and submit a waiver to this GPR in accordance with the requirements of GPR 1400.1, Waiver Processing. This will be done by working with the SSE and/or SWRE.

7.6. The SAE shall report software safety and software reliability process or product non-conformances and risks to the CSO, PSM, SSE, SWRE, RE, and all relevant stakeholders, including the Software PDL.

7.7. The SAE shall review the Software Safety Plan and Software Reliability Plan.

7.8. The SAE shall analyze the software safety requirements and mission-critical software requirements by:

a. Examining the requirements for ambiguities, inconsistencies, omissions, and undefined conditions.

b. Verifying that all safety-critical and mission-critical software requirements are derived from and traceable to system requirements, interface and performance specifications, and/or standards , as well as the system hazard reports or reliability analyses, respectively.

7.9. The SAE shall assure that software design elements that implement safety-critical or mission-critical requirements are designated as safety-critical or mission-critical, respectively.

7.10. The SAE shall assure that all safety-critical and mission-critical software design elements are traceable to and from safety-critical or mission-critical software requirements, respectively.

7.11. The SAE shall assure all safety-critical and mission-critical code is traceable to safety-critical or mission-critical software design elements, respectively.

7.12. The SAE shall assure that all software safety requirements and mission-critical software requirements have been tested, evaluated, inspected, or demonstrated.

7.13. The SAE shall identify all software changes related to safety-critical and mission-critical software. This will be achieved by working with the PSM and SSE or RE and SWRE, respectively.

7.14. The SAE shall evaluate all software changes related to safety-critical and mission-critical software. This will be done by working with the PSM and SSE or RE and SWRE, respectively.

7.15. The SAE shall be a part of any change control board that approves software modifications affecting safety-critical or mission-critical systems. The SAE will work with the SSE or SRE, respectively before approving such modifications.

7.16. The SAE shall evaluate all discrepancy reports related to safety-critical and mission-critical software. This will be accomplished by working with the PSM and SSE or RE and SWRE, respectively.

7.17. The SAE shall approve safety-critical and mission-critical discrepancy report closures and software modifications affecting safety-critical or mission-critical systems. The SAE will work with the SSE or SRE, respectively before approving such closures or modifications.

8. Reliability Engineer

The RE is responsible for addressing software reliability planning as part of the Reliability Program Plan. The RE is responsible for identifying and documenting appropriate software-related failure modes and software-related mitigations (including specifics of fault management) in conducting the FMEA and FTA. The RE works with systems engineering, software engineering, software assurance, project safety and project management to identify reliability risks associated with software.

8.1. The RE is dependent upon overall mission requirements, including the Level 1 Requirements to provide the initial scope for their analysis, as the Mission Classification and Mission Objectives and provide context for establishing key planning parameters for safety and reliability activities, such as:

a. Risk Classification

b. Mission Environments

c. Identification of critical science (vs. technology demonstration efforts)

d. Identification of primary instruments and critical ground systems

e. Mission Success Thresholds

8.2. The RE shall classify software as mission-critical if the software can cause or contribute to or provides a mitigation(s) to a mission failure commensurate with Level 1 and/or threshold science requirements.

8.3. The RE shall assure:

a. All software-related failure modes and faults have been identified.

b. All software-related mitigations have been identified.

c. All mission-critical software requirements and associated software design elements have been identified and tracked.

d. All mission-critical software requirements and associated software design elements have been successfully validated.

e. All mission-critical software requirements and associated software design elements have been properly verified.

f. All discrepancies in mission-critical software have been dispositioned.

8.4. The RE shall evaluate all off-the-shelf and reused software, including software present in Ground Support Equipment for potential reliability impacts.

8.5. The RE shall ensure that the acquired or developed system is evaluated for impacts to software-related failure modes and faults and software-related mitigations resulting from changes to the associated hardware or software (e.g., change requests, discrepancy reports, waivers, problem reports, etc.).

8.6. The RE shall ensure that the above-mentioned evaluations are performed (at a minimum) at major lifecycle reviews and when significant changes are made to the system.

8.7. The RE shall communicate software reliability risks directly to the Software PDL and elevate risks to the Systems Engineer and/or the Project Manager, if necessary, for resolution within the project.

8.8. The RE shall, in parallel with the above-mentioned task (section 8.7), communicate software reliability risks directly to the CSO and SMA Management.

8.9. Utilizing the reliability analyses, such as the FMEAs and FTAs, the RE shall provide input to the Software PDL regarding which failures to test for and the level of failure combinations to include.

8.10. The RE shall apply this GPR to all mission-critical software regardless of the presence of non-software mitigations (e.g., operator intervention, hardware overrides).

8.11. The RE may tailor this GPR for mission-critical software that either can cause a critical failure mode or fault, but meets fault tolerance requirements via the use of independent, non-software-related inhibits or aids in controlling a critical failure mode or fault, but only plays a supplementary role to the required fault tolerance strategy.

8.12. Note that the software mentioned in the above paragraph (section 8.11) is likely to be low risk. As a result, even though such software is still considered to be mission-critical and the associated requirements are tagged as such, this GPR may be tailored such that the level of effort for this

software throughout the remainder of the project lifecycle is focused upon ensuring that any changes made to the software or hardware neither allow the software to bypass nor affect the capability of the non-software-related fault tolerance strategy.

8.13. The RE shall coordinate software reliability tasks with the CSO.

8.14. The RE shall provide input to the CSO regarding the funding levels associated with the software reliability tasks.

9. Software Safety Engineer

The SSE is responsible for reviewing the safety analyses and conducting the software safety analyses to ensure that software is appropriately included. The SSE will identify software-related hazard causes and software-related hazard controls to mitigate the hazard to an acceptable level of risk. The SSE provides training and consultation on project software safety issues. If an SSE does not exist for a project, the CSO will work with Code 320 or Code 803 to determine who shall fulfill the responsibilities of the SSE.

9.1. The SSE shall create a project Software Safety Plan. The Software Safety Plan may be included as a part of other documents, such as the System Safety Program Plan, the Software Management Plan or the Software Assurance Plan.

9.2. The SSE shall specify, in the Software Safety Plan or other document(s):

a. The software safety activities to be carried out, the schedule on which they will be implemented, the personnel who will carry out the activities, the methodologies used, and the products that will result.

b. The interrelationships and external agreements among system safety, software safety, software assurance, software engineering, the PDLs and the SMA organization.

9.3. The SSE shall ensure that software safety activities are performed during the software development lifecycle.

9.4. The SSE shall work with the PSM to identify safety-critical software.

9.5. The SSE shall conduct software safety analyses in conjunction with and using input from the overall safety analyses.

9.6. The SSE shall address hazards associated with a software requirement, design concept and/or operation including software-related hazard causes or software-related hazard controls by ensuring:

a. All software-related hazard causes have been identified.

b. All software-related hazard controls have been identified.

c. All software safety requirements and elements have been identified and tracked.

d. All software safety requirements and elements have been successfully validated.

e. All software safety requirements and elements have been properly verified.

f. All discrepancies in safety-critical software have been dispositioned.

9.7. The SSE shall analyze the software safety requirements using the following steps:

a. Assure that the software-related safety requirements include adequate response to potential failures. Areas to consider should include, but are not limited to: limit ranges, relationship logic for interdependent limits, out-of-sequence event protection, timing problems, sensor or actuator failures, voting logic, hazardous command processing requirements, Fault Detection, Isolation, and Recovery (FDIR), switchover logic for failure tolerance, and the ability to reach and maintain a safe state if so required.

b. Assure that the software-related safety requirements properly implement software-related hazard controls.

9.8. The SSE shall ensure bidirectional traceability between safety-critical software requirements and the associated hazard controls and verifications.

9.9. The SSE shall analyze the software design by:

a. Assuring that system interactions do not invalidate software-related hazard controls.

b. Assuring that the design prioritizes safety features over other software functionality

c. Assuring that the design maintains the system in a safe state during all modes of operation.

9.10. The SSE shall analyze the software implementation by:

a. Assuring that code implementation does not compromise any software-related hazard controls

b. Assuring that code implementation maintains the system in a safe state during all modes of operation.

c. Assuring that the code implements all identified software-related hazard controls.

9.11. The SSE shall review test planning documentation to assure that safety testing has been adequately addressed.

9.12. If additional hazardous states or contributors are identified during testing, the SSE shall perform a complete software safety analysis prior to software delivery or use.

9.13. The SSE shall evaluate all software changes related to safety-critical software for potential safety impact. This will be achieved by working with the SAE. The software safety change evaluation should analyze whether the proposed change could invoke a hazardous state, affect a hazard control, increase the likelihood of a hazardous state, adversely affect safety-critical software, or change the safety-criticality of an existing software design element.

9.14. The SSE shall review changes to the system hazard analyses.

9.15. The SSE shall provide information on changes in safety-critical software and software safety analysis results to the PSM for evaluation and incorporation into the hazard analysis.

9.16. The SSE shall assist the PSM in assessing risks associated with potential software-related failures.

10. Software Reliability Engineer

The SWRE is responsible for reviewing the reliability analyses and conducting the software reliability analyses to ensure that software is appropriately included. The SWRE is responsible for identifying software-related failure modes and software-related faults and mitigations to eliminate or control the failures to an acceptable level of risk. The SWRE provides training and consultation on project software reliability issues. If an SWRE does not exist for a project, the CSO will work with Code 320 or Code 803 to determine who shall fulfill the responsibilities of the SWRE.

10.1. The SWRE shall create a Software Reliability Plan consistent with the project risk management approach. The Software Reliability Plan may be included as a part of other documents, such as the Project Reliability Program Plan, the Software Management Plan or the Software Assurance Plan.

10.2. The SWRE shall specify the software reliability activities to be carried out, the schedule on which they will be implemented, the personnel who will carry out the activities, the methodologies used, and the products that will result.

10.3. The SWRE shall ensure that software reliability activities are performed during the software development lifecycle through mission operations and retirement.

10.4. The SWRE shall document the interrelationships among system safety, software reliability, software assurance, software engineering, and the Center or Program SMA organization in the Software Reliability Plan.

10.5. The SWRE shall work with the RE to identify mission-critical software.

10.6. The SWRE shall conduct software reliability analyses in conjunction with the overall reliability analyses.

10.7. The SWRE shall identify loss of mission scenarios associated with a specific requirement, design concept and/or operation including software-related failure modes and faults or software-related mitigations.

10.8. The SWRE shall ensure:

a. All software-related failure modes and faults have been identified.

b. All software-related mitigations have been identified.

c. All mission-critical software requirements and software-related design elements have been identified and tracked.

d. All mission-critical software requirements and software-related design elements have been successfully validated.

e. All mission-critical software requirements and software-related design elements have been properly verified.

f. All discrepancies in mission-critical software have been dispositioned.

10.9. The SWRE shall analyze the mission-critical software requirements using the following steps:

a. Assure that the mission-critical software requirements include adequate response to potential failures. Areas to consider should include, but are not limited to: limit ranges, relationship logic for interdependent limits, out-of-sequence event protection, timing problems, sensor or actuator failures, voting logic, mission-critical command processing requirements, FDIR, switchover logic for failure tolerance, and the ability to reach and maintain a safe state if so required.

b. Assure that the mission-critical software requirements implement software-related failure mitigations.

10.10. The SWRE shall review test planning documentation to assure that reliability testing has been included.

10.11. The SWRE shall trace software-related failure and fault mitigations and associated verifications to mission failures.

10.12. The SWRE shall analyze the software design by:

a. Assuring that system interactions do not invalidate software-related failure mitigations.

b. Assuring that the design prioritizes mission-critical software features over other software functionality post spacecraft separation.

c. Assuring that the design maintains the system in a safe state during all modes of operation.

10.13. The SWRE shall analyze the software implementation by:

a. Assuring that the code implementation does not compromise any failure mitigations.

b. Assuring that the code implementation maintains the system in a safe state during all modes of operation.

c. Assuring that code implements all identified software-related failure mitigations.

10.14. If additional failure modes, faults or contributors are identified during testing, the SWRE shall update the existing software reliability analyses prior to software delivery or use.

10.15. The SWRE shall evaluate all software changes related to mission-critical software for potential reliability impact. The software reliability change evaluation should analyze whether the proposed change could create a failure mode or fault, affect a failure mitigation, increase the likelihood of a failure mode or fault, adversely affect mission-critical software, or change the mission-criticality of an existing software element.

10.16. The SWRE shall review changes to the reliability analyses.

10.17. The SWRE shall provide information on changes in mission-critical software and updates to the software reliability analyses to the RE for evaluation and incorporation into the reliability analyses.

10.18. The SWRE shall assist the RE in assessing risks associated with potential software-related failures.

APPENDIX A – DEFINITIONS

A.1 Assure - To make certain that specified software assurance, management, and engineering activities are performed by others.

A.2 Design Element - A basic component or building block in a design. A design element could include modules, classes, functions, processes, physical machines and so on, as determined by the architecture documentation.

A.3 Ensure - To perform the specified software assurance, management, and engineering activities.

A.4 Failure - Non-performance or incorrect performance of an intended function of a product. A failure is often the manifestation of one or more faults.

A.5 Failure Modes And Effects Analysis (FMEA) - A bottom-up systematic, inductive, methodical analysis performed to identify and document all identifiable failure modes at a prescribed level and to specify the resultant effect of the modes of failure.

A.6 Fault - An inherent defect in a product which may or may not ever manifest, such as a bug in software code.

A.7 Fault Detection, Isolation, and Recovery (FDIR) - The systematic process associated with determining a fault has occurred along with its source or location, and overcoming said fault without permanently affecting the operational capability of the system.

A.8 Fault Tree Analysis (FTA) - An analytical technique, whereby an undesired system state is specified and the system is then analyzed in the context of its environment and operation to find all credible ways in which the undesired event can occur.

A.9 Hazard - Existing or potential condition that can result in, or contribute to, a mishap or accident.

A.10 Hazard Analysis - Identification and evaluation of existing and potential hazards and the recommended mitigation for the hazard sources found.

A.11 Hazard Control - Means of reducing the risk of exposure to a hazard. This includes design or operational features used to reduce the likelihood of occurrence of a hazardous effect or the severity of the hazard.

A.12 Hazard Mitigation - Any action that reduces or eliminates the risk from hazards.

A.13 Independent Verification And Validation (IV&V) - A discipline of Software Assurance that plays a role in the overall NASA software risk mitigation strategy applied throughout the life cycle to improve the safety and quality of software systems. Verification and validation

performed by an organization that is technically, managerially, and financially independent of the development organization.

A.14 Mission-Critical - Item or function that must retain its operational capability to assure mission success.

A.15 Mission-Critical Software - Software that can cause, contribute to, or mitigate the loss of capabilities that are essential to the primary mission objectives. The software reliability assessment and analysis is focused on failure modes specific to post-separation mission phases. Mission-critical software is identified based on the results of the Failure Modes and Effects Analyses (FMEA) and the Probabilistic Risk Assessment (PRA). Examples of mission-critical software can be found in all types of systems, including Flight, Ground Support System, Mission Operations Support Systems, and Test Facilities.

A.16 Off-The-Shelf (OTS) Software

a. Commercial Off-The-Shelf (COTS) - COTS software refers to purchased OTS software such as operating systems, libraries, or applications developed by non-governmental entities for commercial sales.

b. Modified Off-The-Shelf (MOTS) - MOTS software is typically a COTS or GOTS product whose source code can be modified and that has been customized for a particular use.

c. Government Off-The-Shelf (GOTS) - GOTS is OTS software developed by the government and reused by another Project.

A.17 Preliminary Hazard Analysis (PHA) - An analysis technique used for the identification and evaluation of existing and potential hazards and the recommended mitigation for the hazard sources found.

A.18 Reliability Analysis - An evaluation of reliability of a system or portion thereof. Such analysis usually employs mathematical modeling, directly applicable results of tests on system hardware, estimated reliability figures, and non-statistical engineering estimates to ensure that all known potential sources of unreliability have been evaluated.

A.19 Risk - The combination of (1) the probability (qualitative or quantitative) that a program or project will experience an undesired event and (2) the consequences, impact, or severity of the undesired event were it to occur.

A.20 Safety Analysis - Generic term for a family of analyses, which includes but is not limited to, preliminary hazard analysis, system (subsystem) hazard analysis, operating hazard analysis, software hazard analysis, sneak circuit, and others.

A.21 Safety-Critical - Any condition, event, operation, process, equipment, or system that possesses the potential of directly or indirectly causing harm to humans, destruction of the system, damage to property external to the system, or damage to the environment.

A.22 Safety-Critical Software - Software that can cause, contribute to, or mitigate human safety hazards or damage to flight hardware and facilities. The software safety assessment and analysis is focused on hazards specific to Integration and Test, launch, and up through spacecraft separation from the launch vehicle (except for International Space Station (ISS) payloads that have constant human presence) and re-entry/recovery (where applicable). Safety-critical software is identified based on the results of the hazard analysis and the results of the Orbital Debris Assessment Report/End-Of-Mission Plan (where applicable). Examples of safety-critical software can be found in all types of systems, including Flight, Ground Support System, Mission Operations Support Systems, and Test Facilities.

A.23 Software - Computer programs, procedures, rules, and associated documentation and data pertaining to the development and operation of a computer system. Software includes computer programs, procedures, scripts, rules, and associated documentation and data pertaining to the development and operation of a computer system. This also includes COTS software, GOTS software, MOTS software, custom software, reused software, heritage software, auto generated code, code that is executed on microprocessors within digital electronics, firmware and open source software components. Firmware is software that is stored on nonvolatile memory.

A.24 Software Assurance - The planned and systematic set of activities that ensure that software life cycle processes and products conform to requirements, standards, and procedures. For NASA this includes the disciplines of Software Quality (functions of Software Quality Engineering, Software Quality Assurance, and Software Quality Control), Software Safety, Software Reliability, Software Verification and Validation, and IV&V.

A.25 Software-Related Event/Item - Event (e.g., hazard, fault, failure mode or failure) or item (e.g., hazard control, requirement, function or mitigation) whose outcome or desired outcome is inextricably or causally linked to software functionality (i.e., without software, the event or the desired end-state as expressed by the item would not be possible).

A.26 Software Reliability - The discipline of software assurance that (1) defines the requirements for software controlled system fault/failure detection, isolation, and recovery; (2) reviews the software development processes and products for software error prevention and/or reduced functionality states; and (3) defines the process for measuring and analyzing defects and defines/derives the reliability and maintainability factors.

A.27 Software Safety - The aspects of software engineering and software assurance that provide a systematic approach to identifying, analyzing, and tracking software mitigation and control of hazards and hazardous functions (e.g., data and commands) to ensure safer software operation within a system.

A.28 Software Safety Analysis - The application of system safety engineering techniques throughout the software life cycle to ensure that errors that could reduce system safety have been eliminated or controlled to an acceptable level of risk.

A.29 Validation - In design and development, validation is the process of examining the product to determine conformity with the user's functional requirements. This includes product inspections, functional and operational tests, and environmental simulations.

A.30 Verification - In design and development, verification is the process of examining the design to determine conformity with the documented design requirements. This includes reviewing documentation prior to release, performing alternate evaluations to verify the original analysis, and performing physical tests of hardware and operational tests of software.

APPENDIX B – ACRONYMS

COTS	Commercial-Off-The-Shelf
CSO	Chief Safety and Mission Assurance Officer
FDIR	Fault Detection, Isolation, and Recovery
FMEA	Failure Modes and Effects Analysis
FTA	Fault Tree Analysis
GB	Guidebook
GOTS	Government-Off-The-Shelf
GPR	Goddard Procedural Requirements
GSFC	Goddard Space Flight Center
IEEE	Institute of Electrical and Electronics Engineers
ISS	International Space Station
IV&V	Independent Verification And Validation
MAR	Mission Assurance Requirements
MOTS	Modified-Off-The-Shelf
NASA	National Aeronautics and Space Administration
NPR	NASA Procedural Requirements
OTS	Off-The-Shelf
PDL	Product Development Lead
PHA	Preliminary Hazard Analysis
PM	Project Manager
PSM	Project Safety Manager
RE	Reliability Engineer
SMA	Safety and Mission Assurance
SAE	Software Assurance Engineer
SATERN	System for Administration, Training, and Educational Resources at NASA
SDP	Safety Data Package
SE	Systems Engineer
SSE	Software Safety Engineer
STD	Standard
STEP	Safety and Mission Assurance Technical Excellence Program
SwAT	Software Assurance Tracker
SWRE	Software Reliability Engineer
WFF	Wallops Flight Facility

CHANGE HISTORY LOG

Revision	Effective Date	Description of Changes
Baseline	09/24/12	Initial Release